L Number	Hits	Search Text	DB	Time stamp
1	364559	optimiz$5	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/03/18 16:24
2	2649545	head or arm or actuator	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/03/18 16:24
3	7140	optimiz$5 with (head or arm or actuator)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/03/18 16:25
4	3700	optimiz$5 near5 (head or arm or actuator)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/03/18 16:25
5	1496	optimiz$5 near2 (head or arm or actuator)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/03/18 16:26
6	2356136	movement or motion	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/03/18 16:28
7	359	optimiz$5 with (head or arm or actuator) with (movement or motion)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/03/18 16:31
8	126	optimiz$5 near5 (head or arm or actuator) near5 (movement or motion)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/03/18 16:32

US-PAT-NO: 5809295

DOCUMENT-IDENTIFIER: US 5809295 A

TITLE: Method and apparatus for storing compressed file data on a disk where each MDFAT data structure includes an extra byte

--------- KWIC --------

Detailed Description Text - DETX (50):

By contrast, the method of the present invention stores all fragment locations in the first fragment. Thus, the computer can obtain all fragment locations before the command is issued for the head to move to the proper location to access each fragment. With all fragment locations known, the computer can make an optimized order of access of the fragments such that head movements are minimized, and access time is reduced. Therefore, the inventive method of storing the list of fragment locations is more efficient than prior methods.

0

~QR

DOCUMENT-IDENTIFIER: US 5761692 A

TITLE: Method and apparatus of retrieving continuous and non-continuous media data from a file system

-------- KWIC --------

Brief Summary Text - BSTX (14):
In an alternative aspect of the present invention, a disk scheduling process called C-LOOK is implemented. C-LOOK minimizes seek time during data retrievals in the course of a common period by optimizing movement of a disk arm of a disk reader over a disk.

Detailed Description Text - DETX (18):
ER disk scheduling further provides that once the last instance of a request R.sub.i completes in a service cycle, R.sub.i is deleted from the service list in the next service cycle only after instances of requests before R.sub.i in the list have been completed. Delaying the deletion of R.sub.i until the next service cycle in conjunction with equation [5] and the fact that a subsequent service cycle begins immediately after the previous service cycle ends ensures that servicing new requests earlier in a subsequent service cycle does not cause the difference between the start times of two consecutive instances of a request to exceed T. Data, therefore, is available for transfer to a client after the first instance completes, and requests are guaranteed to be adequately serviced or, as referred to conventionally in the art, requests do not starve. While ER yields a worst case response time of T, the average response time will typically be less than T. It is noted that ER provides for servicing of requests with respect to their time of arrival and does not optimize disk arm movement, such that the length of each instance is computed based on worst-case assumptions.

Detailed Description Text - DETX (19):
In another aspect of the present invention, a disk scheduling technique, called C-LOOK, provides for a reduction in the amount of time which must be allocated for the seek time, when multiple requests are serviced within a common period, by optimizing disk arm movement during a service cycle. As more fully discussed below, reducing seek time provides that a larger number of requests can be admitted for servicing within a common period T.

Detailed Description Text - DETX (21):
In C-LOOK, disk arm movement is optimized as follows. Requests are ordered in the service list according to the positions on disk to be accessed in order to provide that the disk arm moves in one direction for servicing requests in a service cycle. For purposes of clarity, C-LOOK is explained at this point based on the assumption that continuous media clips are stored contiguously on disk, such that the relative position of two requests in the service list does not change from one service cycle to another.

0, -QR

DOCUMENT-IDENTIFIER: US 6496899 B1

TITLE: Disk scheduling system with bounded request reordering

filed 2/28/00 ?

——— KWIC ———

Brief Summary Text - BSTX (11):

Many multimedia applications require continuous media streams in which data streams must be delivered at a specified and possibly time-varying data rates and with a specified uniformity of that delivery rate. In some cases, the uniformity of the delivery rate may be adversely affected by the algorithm used to satisfy disk access requests. The use of a "first-come, first-served" disk access algorithm may not always be the most efficient way to satisfy disk requests, as motion of the read-write head (used to access information from the disk) may be less than optimal. Some optimization of head motion may be realized through the use of algorithms that re-order the disk requests. In such re-ordering algorithms, disk requests may be satisfied in an order different from the order in which they were made. One such re-ordering algorithm is known as an "elevator" algorithm. In one typical elevator algorithm, the head of the disk storage system sweeps from the outer disk to the inner disk, satisfying and queued disk request along the way, and then reversing direction. While this algorithm may allow for more efficient motion of the read-write head, highly non-uniform access times may still be present, as newly arriving requests may be satisfied prior to previously queued requests. A large number of newly arriving requests may cause long delays in satisfying previously queued requests.

O, QR

Brief Summary Text - BSTX (14):

The problems outlined above may in large part be solved by a system and method of bounded disk request reordering in accordance with the present invention. In one embodiment, disk access requests may be performed during traversals of a disk head across a disk. Each traversal may have a specified direction of motion. A plurality of disk accesses may be performed during a disk head traversal. In some cases, disk accesses may be performed in an order different from the order in which the original disk access requests were received. The overall number of disk access requests for a given disk head traversal may be limited to a maximum number N. By limiting the number of disk requests for each traversal, a bound may effectively be placed on the amount of time it takes to satisfy any single disk request, despite any reordering. Disk head motion may be optimized as well.

Brief Summary Text - BSTX (17):

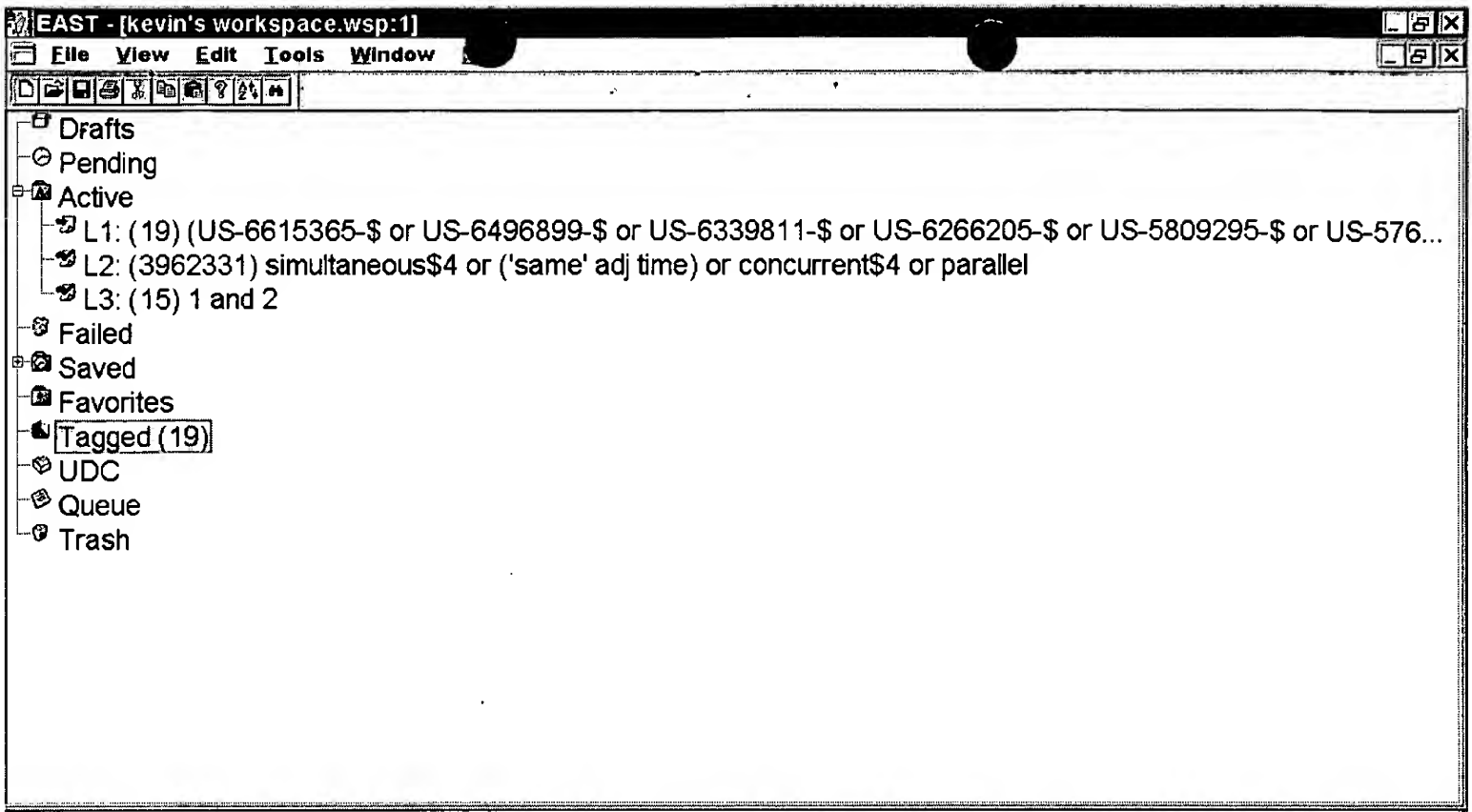
The structure of the algorithms may allow for optimization of disk head motion and more uniform disk access times, despite any reordering. Since the number of disk requests for a given traverse is bounded by a maximum value ("N"), the amount of time to satisfy a given disk request may be bounded as well. In effect, the system utilizes an elevator algorithm with a bounded maximum delay for a given disk request.

Brief Summary Text - BSTX (18):

Thus, in various embodiments, the system and method of bounded disk request reordering may allow disk requests to be reordered and satisfied within specified bounds. This may result in an optimization of disk head motion, and furthermore, allow for more uniform disk access times. The uniformity of disk access times may make the system more suitable for certain applications in which a relatively steady data stream is required. As such, the system may be particularly suited for use with various multimedia applications.

Detailed Description Text - DETX (45):

If the disk block address is beyond the address of the disk head (Step 2004), or the currently searched traversal is not active (active=false, Step 2002), the search algorithm then looks at the number of disk requests in the disk request list (Step 2003). Prior to beginning the search algorithm, a maximum number N of disk requests per disk head traversal is specified. By limiting the number of disk requests per disk head traversal, the response time for a given disk access request may be effectively bounded. This may allow for relatively uniform disk access times, which may be required for certain applications (particularly multimedia applications). Typical values of N are between 8 and 10 requests per traversal, although the value of N may be changed to suit various embodiments. Large values of N typically result in greater optimization of disk head motion, although disk access times may be less uniform. Conversely, smaller values of N may allow for more uniform disk access times, with less optimization of disk head motion.



	U	1	Document ID	Issue Dat	Page	Title	Current OR	Current XR	Retrieval
1	□	☑	WO 200165835 A	20030826	25	Disk storage system sched			
2	□	☑	US 20030079080 A	20030424	27	Disk scheduling system with	711/112	711/167	
3	□	☑	US 6496899 B1	20021217	25	Disk scheduling system with	711/112	711/111;	
4	□	☑	US 20030135522 A	20030717	22	Integrated content manage	707/200		
5	□	☑	US 6339811 B1	20020115	17	Rotationally optimized seek	711/112	711/137	
6	□	☑	JP 09106327 A	19970422	6	DATA TRANSFER SCHED			
7	□	☑	US 5313585 A	19940517	51	Disk drive array with reques	711/201	711/112;	
8	□	☑	US 6615365 B1	20030902	20	Storing a computer disk ima	714/6	711/161;	
9	□	☑	US 5506977 A	19960409	51	Method and controller for mi	711/155		
10	□	☑	US 5619723 A	19970408	51	System for scheduling read	710/3	711/111	
11	□	☑	US 5483641 A	19960109	55	System for scheduling read	710/3	711/111	
12	□	☑	US 5313626 A	19940517	51	Disk drive array with efficie	714/5		
13	□	☑	JP 10027069 A	19980127	13	STORAGE DEVICE			
14	□	☑	US 5761692 A	19980602	23	Method and apparatus of re	711/4	707/205;	
15	□	☑	US 5530960 A	19960625	53	Disk drive controller accepti	710/5		
16	□	☑	US 5473761 A	19951205	49	Controller for receiving tran	711/4	710/26;	
17	□	☑	US 6266205 B1	20010724		Parallel servo with ultra high	360/77.06	360/77.02;	
18	□	☑	US 5809295 A	19980915	24	Method and apparatus for s	707/1	707/101;	
19	□	☑	US 4935825 A	19900619	13	Cylinder defect managemen	360/54	360/53	

DOCUMENT-IDENTIFIER: US 20030135522 A1

TITLE: Integrated content management and block layout technique

-------- KWIC -------

Summary of Invention Paragraph - BSTX (11):
[0010] In addition, in systems with multiple read/write arms, blocks which tend to be accessed together can be spread among the media serviced by these arms such that the blocks can be accessed simultaneously (or nearly simultaneously). The arm scheduler is then responsible for ensuring that the arms are correctly positioned for the concurrent access operations.

Detail Description Paragraph - DETX (28):
[0062] Note that this process relies on a storage system's tendency to store the blocks of a particular file contiguously, when possible. Whereas prior art file systems pass an individual file to a storage system, which uses a layout optimizer to determine how to allocate storage for multiple blocks of that one file (see FIG. 5), the present invention enables allocation to consider cross-file dependencies when allocating storage, as has been described. In this manner, both intra-file allocation (as in the prior art) and inter-file allocation (according to the present invention) can be optimized. While the invention is described with reference to a file system in which read access is optimized by laying out blocks contiguously, some storage systems benefit from other layouts. (For example, multi-arm systems may operate more efficiently when the content is positioned such that the arms can read the content concurrently. This technique is sometimes referred to in the art as "striping".) It will be apparent to one of skill in the art how the techniques which have been disclosed herein can be applied to such systems.

Detail Description Paragraph - DETX (29):
[0063] FIG. 13 illustrates logic which may be used to implement the optional aspect whereby read requests may result in pre-fetching content. A value referred to herein as a "pre-fetch threshold" is used. If a file which is identified in a read request has dependent objects whose dependency weight is greater than the pre-fetch threshold, then in this aspect, additional read requests will be issued for these objects as well. With reference to the example in FIG. 9, suppose for purposes of illustration that the content for file "x.html" has been stored in file storage blocks 1001, 1002, and 1003, which are contiguous, and in block 1006 which is not. Further suppose that the (highly-dependent) image in file "x.jpg" has been stored in contiguous storage blocks 2001 through 2005 and in non-contiguous storage block 2010. Encountering a read request for file "x.html", a prior art file system might recognize the need to read the entire file, and thus execute two parallel read operations for (1) blocks 1001-1003 and (2) block 1006. However, this prior art approach would miss the relationship to the embedded image "x.jpg".

TITLE: Disk scheduling system with bounded request reordering

------- KWIC ------

Pre-Grant Publication Document Identifier - DID (1):
US 20030079080 A1

Summary of Invention Paragraph - BSTX (11):
[0010] Many multimedia applications require continuous media streams in which data streams must be delivered at a specified and possibly time-varying data rates and with a specified uniformity of that delivery rate. In some cases, the uniformity of the delivery rate may be adversely affected by the algorithm used to satisfy disk access requests. The use of a "first-come, first-served" disk access algorithm may not always be the most efficient way to satisfy disk requests, as motion of the read-write head (used to access information from the disk) may be less than optimal. Some optimization of head motion may be realized through the use of algorithms that re-order the disk requests. In such re-ordering algorithms, disk requests may be satisfied in an order different from the order in which they were made. One such re-ordering algorithm is known as an "elevator" algorithm. In one typical elevator algorithm, the head of the disk storage system sweeps from the outer disk to the inner disk, satisfying and queued disk request along the way, and then reversing direction. While this algorithm may allow for more efficient motion of the read-write head, highly non-uniform access times may still be present, as newly arriving requests may be satisfied prior to previously queued requests. A large number of newly arriving requests may cause long delays in satisfying previously queued requests.

Summary of Invention Paragraph - BSTX (14):
[0012] The problems outlined above may in large part be solved by a system and method of bounded disk request reordering in accordance with the present invention. In one embodiment, disk access requests may be performed during traversals of a disk head across a disk. Each traversal may have a specified direction of motion. A plurality of disk accesses may be performed during a disk head traversal. In some cases, disk accesses may be performed in an order different from the order in which the original disk access requests were received. The overall number of disk access requests for a given disk head traversal may be limited to a maximum number N. By limiting the number of disk requests for each traversal, a bound may effectively be placed on the amount of time it takes to satisfy any single disk request, despite any reordering. Disk head motion may be optimized as well.

Summary of Invention Paragraph - BSTX (17):
[0015] The structure of the algorithms may allow for optimization of disk head motion and more uniform disk access times, despite any reordering. Since the number of disk requests for a given traverse is bounded by a maximum value ("N"), the amount of time to satisfy a given disk request may be bounded as well. In effect, the system utilizes an elevator algorithm with a bounded maximum delay for a given disk request.

Summary of Invention Paragraph - BSTX (18):
[0016] Thus, in various embodiments, the system and method of bounded disk request reordering may allow disk requests to be reordered and satisfied within specified bounds. This may result in an optimization of disk head motion, and furthermore, allow for more uniform disk access times. The uniformity of disk access times may make the system more suitable for certain applications in which a relatively steady data stream is required. As such, the system may be particularly suited for use with various multimedia applications.

Detail Description Paragraph - DETX (47):
[0079] If the disk block address is beyond the address of the disk head (Step 2004), or the currently searched traversal is not active (active=false, Step 2002), the search algorithm then looks at the number of disk requests in the disk request list (Step 2003). Prior to beginning the search algorithm, a maximum number N of disk requests per disk head traversal is specified. By limiting the number of disk requests per disk head traversal, the response time for a given disk access request may be effectively bounded. This may allow for relatively uniform disk access times, which may be required for certain applications (particularly multimedia applications). Typical values of N are between 8 and 10 requests per traversal, although the value of N may be changed to suit various embodiments. Large values of N typically result in greater optimization of disk head motion, although disk access times may be less

DERWENT-ACC-NO: 2002-267867

DERWENT-WEEK: 200357

TITLE: Disk storage system schedules access requests to traversals of disk head

INVENTOR: DEMONEY, M A

PATENT-ASSIGNEE: SUN MICROSYSTEMS INC[SUNM]

PRIORITY-DATA: 2000US-0514485 (February 28, 2000), 2002US-0320027 (December 16, 2002)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
JP 2003525486 W	August 26, 2003	N/A	067	G06F 003/06
WO 200165835 A2	September 7, 2001	E	035	H04N 005/00
AU 200145457 A	September 12, 2001	N/A	000	H04N 005/00
EP 1262062 A2	December 4, 2002	E	000	H04N 005/00
US 6496899 B1	December 17, 2002	N/A	000	G06F 013/18
US 20030079080 A1	April 24, 2003	N/A	000	G06F 012/00
KR 2003001367 A	January 6, 2003	N/A	000	H04N 005/781

DESIGNATED-STATES: AE AG AL AM AT AU AZ BA BB BG BR BY BZ CA CH CN CR CU CZ DE DK DM DZ EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT TZ UA UG UZ VN YU ZA ZW AT BE CH CY DE DK EA ES FI FR GB GH GM GR IE IT KE LS LU MC MW MZ NL OA PT SD SE SL SZ TR TZ UG ZW AL AT BE CH CY DE DK ES FI FR GB GR IE IT LI LT LU LV MC MK NL PT RO SE SI TR

APPLICATION-DATA:

PUB-NO	APPL-DESCRIPTOR	APPL-NO	APPL-DATE
JP2003525486W	N/A	2001JP-0563524	February 28, 2001
JP2003525486W	N/A	2001WO-US07091	February 28, 2001
JP2003525486W	Based on	WO 200165835	N/A
WO 200165835A2	N/A	2001WO-US07091	February 28, 2001
AU 200145457A	N/A	2001AU-0045457	February 28, 2001
AU 200145457A	Based on	WO 200165835	N/A
EP 1262062A2	N/A	2001EP-0918372	February 28, 2001
EP 1262062A2	N/A	2001WO-US07091	February 28, 2001
EP 1262062A2	Based on	WO 200165835	N/A
US 6496899B1	N/A	2000US-0514485	February 28, 2000
US20030079080A1	Cont of	2000US-0514485	February 28, 2000
US20030079080A1	N/A	2002US-0320027	December 16, 2002
US20030079080A1	Cont of	US 6496899	N/A
KR2003001367A	N/A	2002KR-0711030	August 23, 2002

INT-CL (IPC): G06F003/06, G06F012/00, G06F013/18, G11B020/10, H04N005/00, H04N005/781, H04N005/93

ABSTRACTED-PUB-NO: WO 200165835A

BASIC-ABSTRACT:

NOVELTY - System comprises a disk, disk head and a scheduler scheduling disk access requests to a first traversal of the disk head and remaining access requests to additional travels. The system maintains a list of disk head traversals including entries

DETAILED DESCRIPTION - There is an INDEPENDENT CLAIM for a method of scheduling disk access requests.

USE - System is for digital video-audio storage and playback systems supporting multiple continuous media streams e.g. multimedia servers.

ADVANTAGE - System limits the number of disk requests for each traversal, limiting request time, and optimizes disk head motion.

DESCRIPTION OF DRAWING(S) - The figure is a flow chart of scheduling of disk access requests in a traversal list.

CHOSEN-DRAWING: Dwg.14/16

DOCUMENT-IDENTIFIER: US 20030079080 A1

TITLE: Disk scheduling system with bounded request reordering

------ KWIC ------

Summary of Invention Paragraph - BSTX (9):
[0008] Even with the use of compression techniques, multimedia applications may still require extremely large amounts of storage. For example, two hours of video encoded at 1 Mb per second may require roughly one gigabyte (1 GB) of storage. A system supporting numerous different content may require up to several terabytes (TB) of storage. The server system must also be able to provide enough bandwidth for the various users to access selected multimedia content without overloading the storage system. For example, to support 100 simultaneous subscribers viewing multimedia content encoded at 1 Mb per second, a server may need to support a bandwidth in excess of 100 Mb per second when allowing for overhead. If enough bandwidth is not available, then some requests may have to be denied, or the play quality may suffer (video may run too slowly or may appear "jerky"). To meet such storage and bandwidth needs, a multimedia server may utilize one or more RAID (Redundant Array of Inexpensive Drives) storage systems. In a RAID system, for a given multimedia file, blocks of multimedia data may be stored across multiple hard disk units. The blocks may be read out or transferred to the communication network and transmitted or broadcast to the user or users. At the receiving end the blocks may be decoded for user viewing on a display device.

Brief Description of Drawings Paragraph - DRTX (8):
[0023] FIG. 5 illustrates one example of a constant data, variable time rate-independent placement mechanism of the video storage manager for two simultaneous continuous media streams;

Detail Description Paragraph - DETX (10):
[0043] Turning briefly to FIG. 5, one example of the constant data, variable time rate-independent placement mechanism of the video storage manager is illustrated for two simultaneous continuous media streams. As shown, the data block size is fixed for all media streams, but the time at which a data block is accessed varies for each stream according to the desired bit rate.

Detail Description Paragraph - DETX (28):
[0060] where N is the number of buffers in the ring and buff_time is a minimum time in which the requester can consume a buffer without exceeding it contracted rate guarantee. Simultaneously with guaranteed rate request being queued with the appropriate disk scheduler 408, prioritized but non-guaranteed rate request are also queued. Non-guaranteed rate request do not carry deadlines but do carry priorities. The disk schedulers issue the queued requests to the storage systems in an order which meets the deadlines associated with the requests while obtaining a high proportion of the disk system bandwidth and allocating residual disk bandwidth after guaranteed requests to non-guaranteed requests in a manner consistent with their priorities.

Detail Description Paragraph - DETX (31):
[0063] Each seek reorder queue 750 is concurrently traversed continuously in one direction (i.e., in increasing or decreasing disk addresses) until no further entries exist in the queue in that direction and it then reverses direction and resumes. Thus, the disk scheduler issues requests from the seek reorder queue to the storage system in order of disk addresses and advances to the next request when the previously issued request has been completed by the disk system.

Detail Description Paragraph - DETX (33):
[0065] Turning now to FIG. 11, a flow chart is provided illustrating operation of the seek reorder queue 750. As indicated at 1102, when the seek reorder queue is not full, a request is migrated from either the deadline or priority queue according to the current cycle slot. If the indicated queue is empty, the request is taken from the alternate queue if that queue is non-empty as indicated at 1104. The migrated request is inserted into the seek reorder queue according to the disk address of the requested block so that requests in the seek reorder queue are ordered by increasing or decreasing disk addresses. Simultaneously, the seek reorder queue is traversed in one direction and the next request is issued to the disk system as indicated at 1108. If the end of

US-PAT-NO: 6496899

DOCUMENT-IDENTIFIER: US 6496899 B1

TITLE: Disk scheduling system with bounded request reordering

------ KWIC -----

Brief Summary Text - BSTX (9):
Even with the use of compression techniques, multimedia applications may still require extremely large amounts of storage. For example, two hours of video encoded at 1 Mb per second may require roughly one gigabyte (1 GB) of storage. A system supporting numerous different content may require up to several terabytes (TB) of storage. The server system must also be able to provide enough bandwidth for the various users to access selected multimedia content without overloading the storage system. For example, to support 100 simultaneous subscribers viewing multimedia content encoded at 1 Mb per second, a server may need to support a bandwidth in excess of 100 Mb per second when allowing for overhead. If enough bandwidth is not available, then some requests may have to be denied, or the play quality may suffer (video may run too slowly or may appear "jerky"). To meet such storage and bandwidth needs, a multimedia server may utilize one or more RAID (Redundant Array of Inexpensive Drives) storage systems. In a RAID system, for a given multimedia file, blocks of multimedia data may be stored across multiple hard disk units. The blocks may be read out or transferred to the communication network and transmitted or broadcast to the user or users. At the receiving end the blocks may be decoded for user viewing on a display device.

Drawing Description Text - DRTX (7):
FIG. 5 illustrates one example of a constant data, variable time rate-independent placement mechanism of the video storage manager for two simultaneous continuous media streams;

Detailed Description Text - DETX (9):
Turning briefly to FIG. 5, one example of the constant data, variable time rate-independent independent placement mechanism of the video storage manager is illustrated for two simultaneous continuous media streams. As shown, the data block size is fixed for all media streams, but the time at which a data block is accessed varies for each stream according to the desired bit rate.

Detailed Description Text - DETX (26):
where N is the number of buffers in the ring and buff time is a minimum time in which the requester can consume a buffer without exceeding it contracted rate guarantee. Simultaneously with guaranteed rate request being queued with the appropriate disk scheduler 408, prioritized but non-guaranteed rate request are also queued. Non-guaranteed rate request do not carry deadlines but do carry priorities. The disk schedulers issue the queued requests to the storage systems in an order which meets the deadlines associated with the requests while obtaining a high proportion of the disk system bandwidth and allocating residual disk bandwidth after guaranteed requests to non-guaranteed requests in a manner consistent with their priorities.

Detailed Description Text - DETX (29):
Each seek reorder queue 750 is concurrently traversed continuously in one direction (i.e., in increasing or decreasing disk addresses) until no further entries exist in the queue in that direction and it then reverses direction and resumes. Thus, the disk scheduler issues requests from the seek reorder queue to the storage system in order of disk addresses and advances to the next request when the previously issued request has been completed by the disk system.

Detailed Description Text - DETX (31):
Turning now to FIG. 11, a flow chart is provided illustrating operation of the seek reorder queue 750. As indicated at 1102, when the seek reorder queue is not full, a request is migrated from either the deadline or priority queue according to the current cycle slot. If the indicated queue is empty, the request is taken from the alternate queue if that queue is non-empty as indicated at 1104. The migrated request is inserted into the seek reorder queue according to the disk address of the requested block so that requests in the seek reorder queue are ordered by increasing or decreasing disk addresses. Simultaneously, the seek reorder queue is traversed in one direction and the

Brief Summary Text - BSTX (177):
Information can also be stored in a stripe across the disk drives of a RAID system; this allows parallel read/writes by the disk drives and thereby increases information transfer speed. NCR's booklet "What are Disk Arrays?" (NCR 1990) illustrates various RAID type configurations)..sup.1

Detailed Description Text - DETX (6):
Controller 100 operates as follows. First, the disk drives 110-114 have their spinning platters synchronized by communication on cable 116; FIG. 2 schematically illustrates synchronization in that sectors 120-124 on the platters of disk drives 110-114, respectively, will pass their corresponding read/write heads 130-134 simultaneously. Platters typically spin at 3,600 rpm, so one revolution takes about 16.7 milliseconds. Each sector contains 512 bytes (1/2 KB) of information, and the number of physical sectors forming a single circular track around the platter may be about 40 for each of about 1,000 concentric tracks on the platter. This translates to a read/write rate of about 1 million bytes per second. (Typical disk drives include multiple platters and multiple heads, for increased capacity.)

Detailed Description Text - DETX (15):
The sameness of the results in equations (2) and (1') follows from the definition of the parity. But the use of equation (2) for the computation of the new parity implies that only two reads of old data are used, rather than the three reads of equation (1'). Also, the reads in equation (2) are on disk drives 110 and 111 which are also the disk drives that will be written; thus disk drives 112-114 are free to be separately read or written (in other tracks) simultaneously. In contrast, equation (1') will have reads of disk drives 112-114 and writes of disk drives 110-111; that is, all five disk drives are active. Equation (2) also has an analog which could be used in place of equation (1) for the case of writing new data to sectors 121-123, namely:

Detailed Description Text - DETX (91):
Controls multiple parallel arrays of 2,3,4, or 5 IDE drives each giving a corresponding increase in disk transfer rate.

Detailed Description Text - DETX (92):
In a parallel array one of the drives can be used for parity data giving complete redundancy given a single drive failure.

Detailed Description Text - DETX (97):
Optionally controls the same number of independent drives with parallel, overlapped seeks and data transfers on up to ten drives at one time.

Detailed Description Text - DETX (124):
The DDA firmware is composed of three conceptual layers: the host interface layer, the request processor layer, and the device driver layer (see FIG. 7). The first conceptual layer, the host interface, is made up of the native interface, the AHA interface, and a set of internal initiators. The host interface layer is responsible for initiating all I/O activity in the controller and its three components can be active simultaneously. This coactivity allows the DDA firmware to perform such things as background event logging and rebuilding, but more interestingly, it allows DDA to emulate non-intelligent controllers while simultaneously supporting array monitoring facilities through its native interface. This design does introduce the complication that host software may believe there are two controllers in the system and try to install two device drivers, so DDA's emulation mode is defeatable and a command is provided in the native interface to allow native device drivers to check the emulation mode.

Detailed Description Text - DETX (142):
Multiple I/O Threads DDA supports multiple outstanding I/Os on each logical drive, with operations on separate logical drives occurring concurrently.

Detailed Description Text - DETX (516):
Concurrent native mode requests: Choice: [16, 32, 64, 128]; Default: 16 w/emulation, 64 w/out emulation.

Detailed Description Text - DETX (517):
This specifies the total number concurrent requests that can be submitted

US-PAT-NO: 5619723

DOCUMENT-IDENTIFIER: US 5619723 A

TITLE: System for scheduling read ahead operations if new request is sequential of last n last read requests wherein n is different on independent activities

--------- KWIC --------

Brief Summary Text - BSTX (180):

Sophisticated controllers for expensive disk drives may reorder the requests in queue (disk scheduling) to improve access efficiency; for example, the shortest-seek-time-first method would first service the request in the queue which involves the smallest distance of head movement, whereas the circular scan with rotational optimization method basically moves the head across the platter from outside track to inside track servicing requests in order of smallest head movement except for cases where radially-further-away sectors serviced first would lessen rotational latency. In these disk-scheduling controllers the request queue may be kept in an elevator queue (the requests are ordered as a function of sector radial distance) with each request identified by its associated handle. However, for inexpensive disk drives with an IDE (ATA) or SCSI interface, as would be used in personal computers, the disk drive includes a controller on its circuit board to take care of hardware details such as motor control for head movement but would not include disk scheduling. An IDE disk drive only communicates with the CPU of a personal computer at a logic level rather than at a device level; this limits the CPU from disk scheduling because the IDE interface may include a mapping of the physical disk drive to appear as a different disk drive. For example, the use of 17 sectors per track was common for disk drives installed in IBM PCs in the early 1980s, but some more recent disk drives have used 40 sectors per track or even varying 35 to 49 sectors per track from spindle edge to outside edge; and these higher density disk drives can logically appear to have 17 sectors with multiple read/write heads. Consequently, IDE type disk drives have a problem of inefficient access, and a RAID with IDE disk drives compounds this problem.

0, QR

US-PAT-NO: 5530960

DOCUMENT-IDENTIFIER: US 5530960 A

TITLE: Disk drive controller accepting first commands for accessing composite drives and second commands for individual diagnostic drive control wherein commands are transparent to each other

--------- KWIC --------

Brief Summary Text - BSTX (183):

Sophisticated controllers for expensive disk drives may reorder the requests in queue (disk scheduling) to improve access efficiency; for example, the shortest-seek-time-first method would first service the request in the queue which involves the smallest distance of head movement, whereas the circular scan with rotational optimization method basically moves the head across the platter from outside track to inside track servicing requests in order of smallest head movement except for cases where radially-further-away sectors serviced first would lessen rotational latency. In these disk-scheduling controllers the request queue may be kept in an elevator queue (the requests are ordered as a function of sector radial distance) with each request identified by its associated handle. However, for inexpensive disk drives with an IDE (ATA) or SCSI interface, as would be used in personal computers, the disk drive includes a controller on its circuit board to take care of hardware details such as motor control for head movement but would not include disk scheduling. An IDIE disk drive only communicates with the CPU of a personal computer at a logic level rather than at a device level; this limits the CPU from disk scheduling because the IDE interface may include a mapping of the physical disk drive to appear as a different disk drive. For example, the use of 17 sectors per track was common for disk drives installed in IBM PCs in the early 1980s, but some more recent disk drives have used 40 sectors per track or even varying 35 to 49 sectors per track from spindle edge to outside edge; and these higher density disk drives can logically appear to have 17 sectors with multiple read/write heads. Consequently, IDE type disk drives have a problem of inefficient access, and a RAID with IDE disk drives compounds this problem.

US-PAT-NO: 5506977

DOCUMENT-IDENTIFIER: US 5506977 A

TITLE: Method and controller for minimizing reads during partial stripe write operations to a disk drive

-------- KWIC --------

Brief Summary Text - BSTX (185):

Sophisticated controllers for expensive disk drives may reorder the requests in queue (disk scheduling) to improve access efficiency; for example, the shortest-seek-time-first method would first service the request in the queue which involves the smallest distance of head movement, whereas the circular scan with rotational optimization method basically moves the head across the platter from outside track to inside track servicing requests in order of smallest head movement except for cases where radially-further-away sectors serviced first would lessen rotational latency. In these disk-scheduling controllers the request queue may be kept in an elevator queue (the requests are ordered as a function of sector radial distance) with each request identified by its associated handle. However, for inexpensive disk drives with an IDE (ATA) or SCSI interface, as would be used in personal computers, the disk drive includes a controller on its circuit board to take care of hardware details such as motor control for head movement but would not include disk scheduling. An IDE disk drive only communicates with the CPU of a personal computer at a logic level rather than at a device level; this limits the CPU from disk scheduling because the IDE interface may include a mapping of the physical disk drive to appear as a different disk drive. For example, the use of 17 sectors per track was common for disk drives installed in IBM PCs in the early 1980s, but some more recent disk drives have used 40 sectors per track or even varying 35 to 49 sectors per track from spindle edge to outside edge; and these higher density disk drives can logically appear to have 17 sectors with multiple read/write heads. Consequently, IDE type disk drives have a problem of inefficient access, and a RAID with IDE disk drives compounds this problem.

US-PAT-NO: 5483641

DOCUMENT-IDENTIFIER: US 5483641 A

TITLE: System for scheduling readahead operations if new request is within a proximity of N last read requests wherein N is dependent on independent activities

--------- KWIC --------

Brief Summary Text - BSTX (184):

Sophisticated controllers for expensive disk drives may reorder the requests in queue (disk scheduling) to improve access efficiency; for example, the shortest-seek-time-first method would first service the request in the queue which involves the smallest distance of head movement, whereas the circular scan with rotational optimization method basically moves the head across the platter from outside track to inside track servicing requests in order of smallest head movement except for cases where radially-further-away sectors serviced first would lessen rotational latency. In these disk-scheduling controllers the request queue may be kept in an elevator queue (the requests are ordered as a function of sector radial distance) with each request identified by its associated handle. However, for inexpensive disk drives with an Integrated Drive Electronics IDE (AT attachment design or ATA) or SCSI (small computer systems-interface) interface, as would be used in personal computers, the disk drive includes a controller on its circuit board to take care of hardware details such as motor control for head movement but would not include disk scheduling. An IDE disk drive only communicates with the CPU of a personal computer at a logic level rather than at a device level; this limits the CPU from disk scheduling because the IDE interface may include a mapping of the physical disk drive to appear as a different disk drive. For example, the use of 17 sectors per track was common for disk drives installed in IBM PCs in the early 1980s, but some more recent disk drives have used 40 sectors per track or even varying 35 to 49 sectors per track from spindle edge to outside edge; and these higher density disk drives can logically appear to have 17 sectors with multiple read/write heads. Consequently, IDE type disk drives have a problem of inefficient access, and a RAID with IDE disk drives compounds this problem.

US-PAT-NO: 5473761

DOCUMENT-IDENTIFIER: US 5473761 A

TITLE: Controller for receiving transfer requests for noncontiguous sectors and reading those sectors as a continuous block by interspersing no operation requests between transfer requests

--------- KWIC --------

Brief Summary Text - BSTX (31):

Sophisticated controllers for expensive disk drives may reorder the requests in queue (disk scheduling) to improve access efficiency; for example, the shortest-seek-time-first method would first service the request in the queue which involves the smallest distance of head movement, whereas the circular scan with rotational optimization method basically moves the head across the platter from outside track to inside track servicing requests in order of smallest head movement except for cases where radially-further-away sectors serviced first would lessen rotational latency. In these disk-scheduling controllers the request queue may be kept in an elevator queue (the requests are ordered as a function of sector radial distance) with each request identified by its associated handle. However, for inexpensive disk drives with an IDE (ATA) or SCSI interface, as would be used in personal computers, the disk drive includes a controller on its circuit board to take care of hardware details such as motor control for head movement but would not include disk scheduling. An IDE disk drive only communicates with the CPU of a personal computer at a logic level rather than at a device level; this limits the CPU from disk scheduling because the IDE interface may include a mapping of the physical disk drive to appear as a different disk drive. For example, the use of 17 sectors per track was common for disk drives installed in IBM PCs in the early 1980s, but some more recent disk drives have used 40 sectors per track or even varying 35 to 49 sectors per track from spindle edge to outside edge; and these higher density disk drives can logically appear to have 17 sectors with multiple read/write heads. Consequently, IDE type disk drives have a problem of inefficient access, and a RAID with IDE disk drives compounds this problem.

US-PAT-NO: 5313626

DOCUMENT-IDENTIFIER: US 5313626 A

TITLE: Disk drive array with efficient background rebuilding

-------- KWIC -------

Brief Summary Text - BSTX (31):

Sophisticated controllers for expensive disk drives may reorder the requests in queue (disk scheduling) to improve access efficiency; for example, the shortest-seek-time-first method would first service the request in the queue which involves the smallest distance of head movement, whereas the circular scan with rotational optimization method basically moves the head across the platter from outside track to inside track servicing requests in order of smallest head movement except for cases where radially-further-away sectors serviced first would lessen rotational latency. In these disk-scheduling controllers the request queue may be kept in an elevator queue (the requests are ordered as a function of sector radial distance) with each request identified by its associated handle. However, for inexpensive disk drives with an IDE (ATA) or SCSI interface, as would be used in personal computers, the disk drive includes a controller on its circuit board to take care of hardware details such as motor control for head movement but would not include disk scheduling. An IDE disk drive only communicates with the CPU of a personal computer at a logic level rather than at a device level; this limits the CPU from disk scheduling because the IDE interface may include a mapping of the physical disk drive to appear as a different disk drive. For example, the use of 17 sectors per track was common for disk drives installed in IBM PCs in the early 1980s, but some more recent disk drives have used 40 sectors per track or even varying 35 to 49 sectors per track from spindle edge to outside edge; and these higher density disk drives can logically appear to have 17 sectors with multiple read/write heads. Consequently, IDE type disk drives have a problem of inefficient access, and a RAID with IDE disk drives compounds this problem.

US-PAT-NO: 5313585

DOCUMENT-IDENTIFIER: US 5313585 A

TITLE: Disk drive array with request fragmentation

--------- KWIC -------

Brief Summary Text - BSTX (184):

Sophisticated controllers for expensive disk drives may reorder the requests in queue (disk scheduling) to improve access efficiency; for example, the shortest-seek-time-first method would first service the request in the queue which involves the smallest distance of head movement, whereas the circular scan with rotational optimization method basically moves the head across the platter from outside track to inside track servicing requests in order of smallest head movement except for cases where radially-further-away sectors serviced first would lessen rotational latency. In these disk-scheduling controllers the request queue may be kept in an elevator queue (the requests are ordered as a function of sector radial distance) with each request identified by its associated handle. However, for inexpensive disk drives with an Integrated Drive Electronics IDE (At attachment design or ATA) or SCSI (small computer systems interface) interface, as would be used in personal computers, the disk drive includes a controller on its circuit board to take care of hardware details such as motor control for head movement but would not include disk scheduling. An IDE disk drive only communicates with the CPU of a personal computer at a logic level rather than at a device level; this limits the CPU from disk scheduling because the IDE interface may include a mapping of the physical disk drive to appear as a different disk drive. For example, the use of 17 sectors per track was common for disk drives installed in IBM PCs in the early 1980s, but some more recent disk drives have used 40 sectors per track or even varying 35 to 49 sectors per track from spindle edge to outside edge; and these higher density disk drives can logically appear to have 17 sectors with multiple read/write heads. Consequently, IDE type disk drives have a problem of inefficient access, and a RAID with IDE disk drives compounds this problem.

PAT-NO: JP409106327A

DOCUMENT-IDENTIFIER: JP 09106327 A

TITLE: DATA TRANSFER SCHEDULING METHOD

PUBN-DATE: April 22, 1997

INVENTOR-INFORMATION:
NAME
GOMI, HIROSHI
SHINTANI, YOSHIHIRO

ASSIGNEE-INFORMATION:

NAME	COUNTRY
KK OKI TECHNO SYST LAB	N/A
OKI ELECTRIC IND CO LTD	N/A

APPL-NO: JP07262732

APPL-DATE: October 11, 1995

INT-CL (IPC): G06F003/06, G06F013/00

ABSTRACT:

PROBLEM TO BE SOLVED: To eliminate the useless movement of a head and to optimize read by arranging data transfer requests in the head seaking direction of a storage medium, moving the request of high priority to the front and reading and transferring data.

SOLUTION: The various kinds of the data requesting a real time property such as moving pictures and sound, etc., are stored in a hard disk 2, the data transfer requests from plural users are received through a network and a head 1 is moved in the direction of the arrow A (seaking direction) and reads the data. In this case, the successively received data transfer requests are held and the requests are successively arranged in the head seaking direction of the hard disk 2. At the time, when response time for an access request is limited within the prescribed time for instance, the priority is set in the access request, the data transfer request is moved in order in the front and the read and transfer of the data are preferentially executed.

0, 8R

DOCUMENT-IDENTIFIER: JP 10027069 A

TITLE: STORAGE DEVICE

PUBN-DATE: January 27, 1998

INVENTOR-INFORMATION:
NAME
OKUMURA, TOMOHIRO
YAMAMOTO, AKIRA
TSUBOI, TOSHIAKI
SHIMIZU, HIROSHI
UCHIYAMA, YOSHIHIRO
TAKAMOTO, KENICHI
MURAOKA, KENJI

ASSIGNEE-INFORMATION:
NAME COUNTRY
HITACHI LTD N/A
HITACHI MICROCOMPUT SYST LTD N/A
HITACHI SOFTWARE ENG CO LTD N/A

APPL-NO: JP08178994

APPL-DATE: July 9, 1996

INT-CL (IPC): G06F003/06, G06F003/06 , G06F012/08 , G06F012/08

ABSTRACT:

PROBLEM TO BE SOLVED: To improve the data read-out processing ability of a storage device when the device receives a plurality of sequential read requests from a host device.

SOLUTION: When a plurality of sequential read requests is made to a host device 103 which functions as a server from a plurality of client devices 101 connected through a LAN 10, the preread control section 143 and preread executing section 148 of the control section 140 of the subordinate storage device 110 of the hose device 103 control a plurality of preread by analyzing a command from the device 103 by using a preread management table which is set on a memory 145 for storing the information used for managing the plurality of preread. When the data on a lower-rank storing medium 180 are transferred to a buffer memory 160, a plurality of reading operations is collectively executed so that the movement of a head on each storing medium 180 is optimized and, when a read request is made to a transferred area, the data are directly transferred to the host device 103 from the memory 160.

US-PAT-NO: 4935825

DOCUMENT-IDENTIFIER: US 4935825 A

TITLE: Cylinder defect management system for data storage system

--------- KWIC --------

Brief Summary Text - BSTX (7):

A primary objective of a disk controller is to minimize the access time of data transfers to and from a disk. Disk access times are not only a function of the physical characteristics of each particular disk, but also of the controller's ability to efficiently organize the format of the disk's data storage areas and to optimize movements of the disk's read/write heads. A controller accomplishes these tasks by segmenting a disk into logical components and coordinating data reads and writes so as to minimize disk access times. The performance of any data processing system is increased by reduced disk access times, resulting in faster retrieval and through-put of data to a host computer.

US-PAT-NO: 6266205

DOCUMENT-IDENTIFIER: US 6266205 B1

TITLE: Parallel servo with ultra high bandwidth off-track detection

--------- KWIC --------

Brief Summary Text - BSTX (11):

All actuator arms 16 in a multiple disk storage device are ganged together so that they move in unison with respect to the disk 4. The actuator arms 16 perform coarse positioning of the transducer 10, while the microactuator arms 13 perform fine position adjustments so that the transducer 10 is centered over a track 25 (see FIG. 1). As shown in FIG. 1, each microactuator arm 13 is pivotally connected to its respective actuator arm 16 and is capable of pivotable movement independent from the actuator arm 16, which allows for fine position adjustments. Movement of each microactuator arm 13 can be independently optimized for imperfections in the arcuate geometry of each track 25 on its corresponding magnetic surface 5. Although FIGS. 1 and 2 depict a transducer positioning system which contains both actuators and microactuators, more commonly, the combination of both positioning methods are not used in a given hard drive 1.

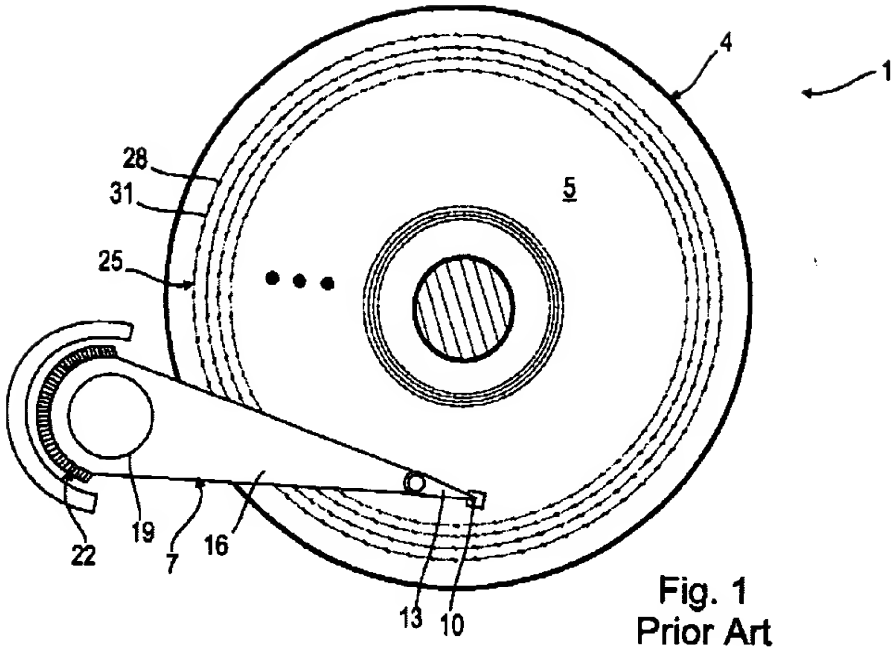


Fig. 1
Prior Art

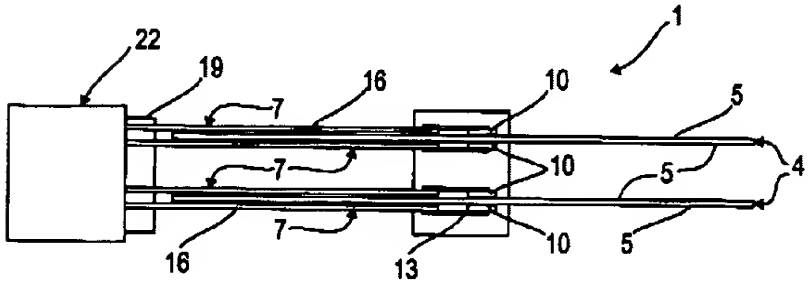


Fig. 2
Prior Art

US-PAT-NO: 6339811

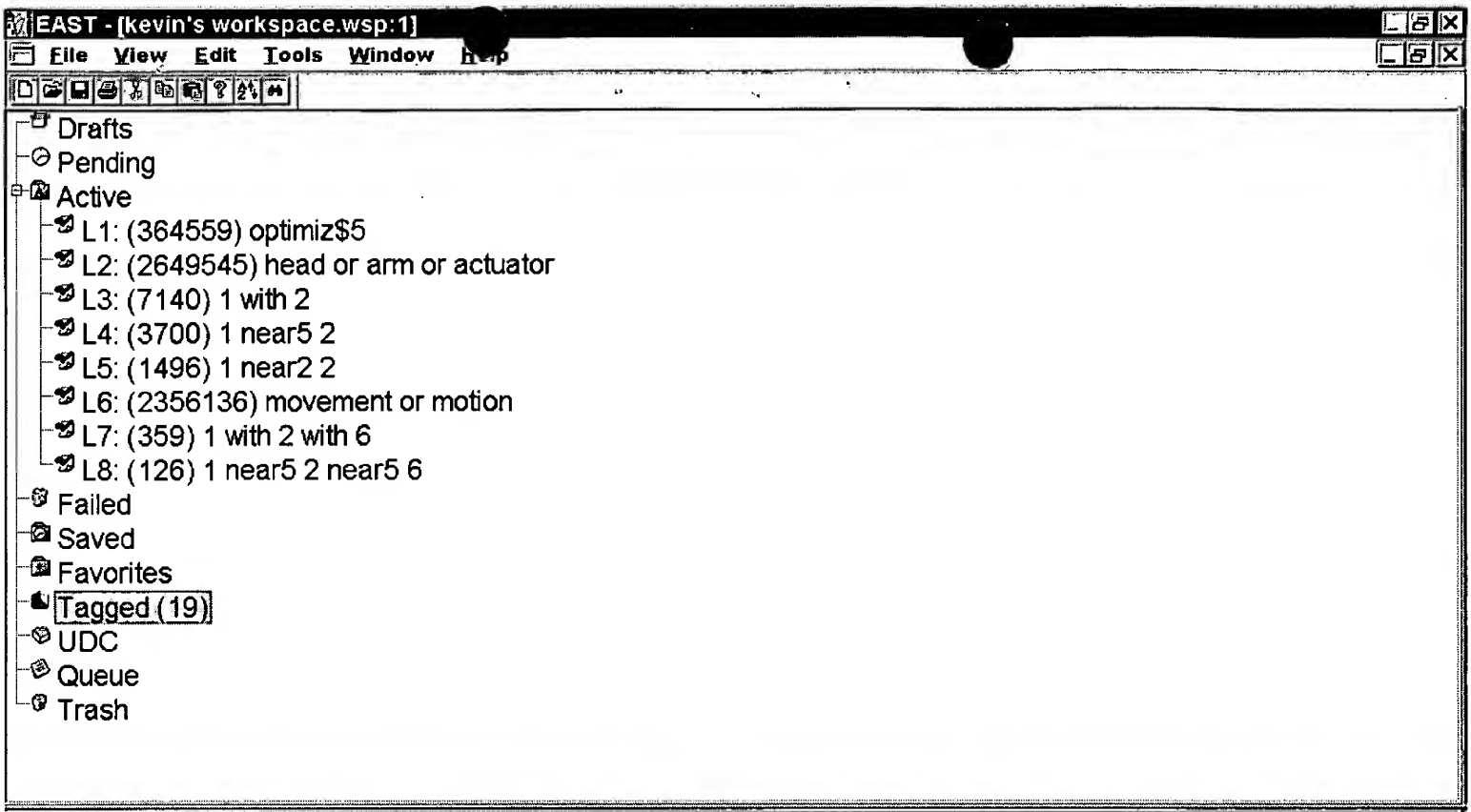
DOCUMENT-IDENTIFIER: US 6339811 B1
See image for Certificate of Correction

TITLE: Rotationally optimized seek initiation

--------- KWIC --------

Detailed Description Text - DETX (2):
A disc drive contains many elements that cooperate to provide data to a host upon request. Among these elements, a control system moves an actuator which contains the read/write head. Control system embodiments of the present invention rotationally optimize the seek initiation of the actuator arm by delaying the actuator movement until the last moment so that the actuator arrives at the new track just in time to begin reading the target data. Seek optimization methods delay the actuator seek initiation by first calculating an amount of rotation or time to delay the seek initiation or by finding a seek initiation trigger. The rotational seek optimization can be applied in the case where each command is executed before the next is received or in cases where several commands are queued and scheduled before being executed.

no reorder



	U	1	Document ID	Issue Dat	Page	Title	Current OR	Current XR	Retrieval
1	☐	☑	WO 200165835 A	20030826	25	Disk storage system sched			
2	☐	☑	US 20030079080 A	20030424	27	Disk scheduling system with	711/112	711/167	
3	☐	☑	US 6496899 B1	20021217	25	Disk scheduling system with	711/112	711/111;	
4	☐	☑	US 20030135522 A	20030717	22	Integrated content manage	707/200		
5	☐	☑	US 6339811 B1	20020115	17	Rotationally optimized seek	711/112	711/137	
6	☐	☑	JP 09106327 A	19970422	6	DATA TRANSFER SCHED			
7	☐	☑	US 5313585 A	19940517	51	Disk drive array with reques	711/201	711/112;	
8	☐	☑	US 6615365 B1	20030902	20	Storing a computer disk ima	714/6	711/161;	
9	☐	☑	US 5506977 A	19960409	51	Method and controller for mi	711/155		
10	☐	☑	US 5619723 A	19970408	51	System for scheduling read	710/3	711/111	
11	☐	☑	US 5483641 A	19960109	55	System for scheduling read	710/3	711/111	
12	☐	☑	US 5313626 A	19940517	51	Disk drive array with efficie	714/5		
13	☐	☑	JP 10027069 A	19980127	13	STORAGE DEVICE			
14	☐	☑	US 5761692 A	19980602	23	Method and apparatus of re	711/4	707/205;	
15	☐	☑	US 5530960 A	19960625	53	Disk drive controller accepti	710/5		
16	☐	☑	US 5473761 A	19951205	49	Controller for receiving tran	711/4	710/26;	
17	☐	☑	US 6266205 B1	20010724	19	Parallel servo with ultra high	360/77.06	360/77.02;	
18	☐	☑	US 5809295 A	19980915	24	Method and apparatus for s	707/1	707/101;	
19	☐	☑	US 4935825 A	19900619	13	Cylinder defect managemen	360/54	360/53	